



## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

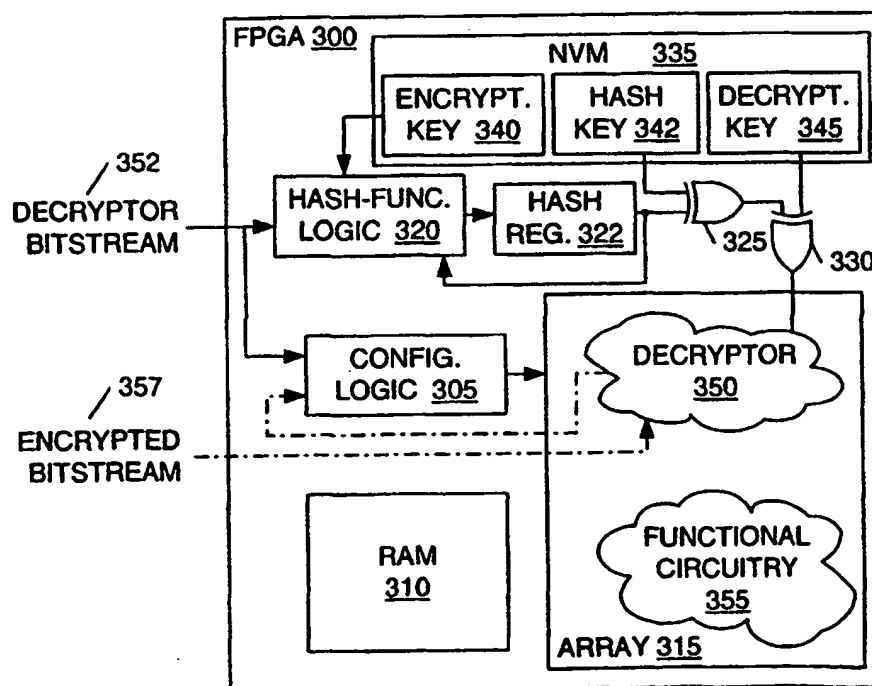
<b>(51) International Patent Classification <sup>7</sup> :</b> <b>H03K 19/00</b>	<b>A2</b>	<b>(11) International Publication Number:</b> <b>WO 00/49717</b> <b>(43) International Publication Date:</b> 24 August 2000 (24.08.00)
<b>(21) International Application Number:</b> PCT/US00/01398 <b>(22) International Filing Date:</b> 19 January 2000 (19.01.00)  <b>(30) Priority Data:</b> 09/253,401      19 February 1999 (19.02.99)      US  <b>(71) Applicant:</b> XILINX, INC. [US/US]; 2100 Logic Drive, San Jose, CA 95124 (US). <b>(72) Inventor:</b> TRIMBERGER, Stephen, M.; 1261 Chateau Drive, San Jose, CA 95120 (US). <b>(74) Agents:</b> YOUNG, Edel, M. et al.; Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 (US).		<b>(81) Designated States:</b> JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).  <b>Published</b> <i>Without international search report and to be republished upon receipt of that report.</i>

**(54) Title:** METHOD AND APPARATUS FOR PROTECTING PROPRIETARY CONFIGURATION DATA FOR PROGRAMMABLE LOGIC DEVICES

**(57) Abstract**

Described are a method of programming a programmable logic device using encrypted configuration data and a programmable logic device (PLD) adapted to use such encrypted data. A PLD is adapted to include a decryptor having access to a non-volatile memory element programmed with a secret decryption key. Some or all of the decryptor can be instantiated in configurable logic on the FPGA. Encrypted configuration data representing some desired circuit functionality is presented to the decryptor. The decryptor then decrypts the configuration data, using the secret decryption key, and configures the FPGA with the decrypted configuration data. Some embodiments include

authentication circuitry that performs a hash function on the configuration data used to instantiate the decryptor on the PLD. The result of the hash function is compared to a proprietary hash key programmed into the PLD. Only those configuration data that produce the desired hash result will instantiate decryptors that have access to the decryption key.



**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TC	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

1           METHOD AND APPARATUS FOR PROTECTING PROPRIETARY  
CONFIGURATION DATA FOR PROGRAMMABLE LOGIC DEVICES

FIELD OF THE INVENTION

6           This invention relates generally to programmable logic  
devices, and in particular to methods and apparatus for  
encrypting data used to configure programmable logic  
devices to protect that data from theft.

BACKGROUND

11           Figure 1 depicts an example of a chip set 100 that  
includes some general-purpose read-only memory (ROM) 110  
connected to a general-purpose FPGA 120. FPGA 120  
conventionally includes an array 130 that can be configured  
to implement custom functional circuitry 140. Array 130 is  
16 typically an array of configurable logic blocks (CLBs)  
programmably interconnected to each other and to  
programmable input/output blocks (IOBs).

          A vendor may use a chip set similar to chip set 100 to  
supply any number of different circuit designs while  
21 stocking only a single general-purpose FPGA and some  
general-purpose memory. The vendor supplies a customer with  
a custom version of chip set 100 by simply programming ROM  
110 with the configuration data required to implement the  
customer's desired function.

26           Configuration data are typically downloaded into an  
FPGA (or other type of programmable logic device) as a  
series of bits known as a configuration bitstream. Anyone  
having access to the configuration bitstream for a  
particular design can easily copy the design. In the  
31 foregoing example in which a vendor sells a custom circuit  
as a set of configuration data combined with a general-  
purpose FPGA, an unscrupulous customer could easily copy  
the configuration data and use it to program any number of  
additional FPGAs. Designs may also be stolen by reverse  
36 engineering the design from the configuration bitstream and  
then adapting the design for another FPGA or even a

1 different circuit technology. Naturally, developers of custom configuration data for use in programmable chips sets are concerned for the security of their designs.

6 Some customers develop their own circuit designs and implement them on FPGAs. Designing complex circuits from basic logic gates, or "primitive cells," can be very time consuming. More complex functions called macros, or "cores," are therefore developed to represent more complex logic functions. These cores can then be used as building blocks for assembling yet more complex circuit designs.

11 A number of core developers design and market cores for FPGAs and other types of programmable logic devices (PLDs). Customers purchase these cores and use them to program PLDs to achieve desired functions. For example, a collection of cores for implementing standard bus  
16 interfaces and signal-processing functions is available from Xilinx, Inc., of San Jose, California, under the name LogiCORE™. As with the configuration data in the example of Figure 1, PLD cores and circuit designs that employ them are easily stolen. Core developers are therefore concerned  
21 for the security of their cores. There is therefore a need for a means of securing cores and other proprietary configuration data.

#### SUMMARY

26 The present invention is directed to a method of configuring a programmable logic device using encrypted configuration data, and to a programmable logic device adapted to use such encrypted configuration data.

31 In one embodiment, a type of programmable logic device commonly known as a field-programmable gate array (FPGA) is adapted to include a decryptor and a non-volatile memory element programmed with a secret decryption key. Some or all of the decryptor can be instantiated in configurable logic on the FPGA. Once the decryptor is instantiated,  
36 encrypted configuration data representing some desired circuit function is presented to the decryptor. The

1 decryptor then decrypts the configuration data, using the  
secret decryption key, and configures the FPGA with the  
decrypted configuration data.

For implementations in which the decryptor is  
instantiated in configuration memory of the FPGA, a clever  
6 thief might engineer an FPGA design that, when  
instantiated, simply reads the decryption key and presents  
the key on an output pin of the FPGA. To forestall such a  
security breach, an FPGA in accordance with a second  
embodiment of the invention includes authentication  
11 circuitry that performs a hash function on the  
configuration data used to instantiate the decryptor. The  
result of the hash function is compared to a proprietary  
hash key programmed into a second non-volatile memory  
element on the FPGA. Only those decryptors whose  
16 configuration data produce the desired hash result will  
have access to the decryption key.

#### BRIEF DESCRIPTION OF THE FIGURES

Figure 1 depicts an example of a conventional chip set  
21 100 that includes some general-purpose read-only memory  
(ROM) 110 connected to a general-purpose FPGA 120.

Figure 2 is a block diagram of an FPGA 200 in  
accordance with an embodiment of the present invention.

Figure 3 is a block diagram of an FPGA 300 in  
26 accordance with another embodiment of the present  
invention.

Figure 4 is a flowchart 400 depicting the process of  
programming FPGA 300 of Figure 3 to include a decryptor and  
some functional circuitry.

31 Figure 5 is a flowchart 500 summarizing the  
conventional Data Encryption Standard (DES) encryption  
algorithm.

Figure 6A is a block diagram 600 representing the hash  
function performed by hash-function logic 320 of Figure 3.

36 Figure 6B is a flowchart 650 illustrating a method of  
performing the hash function of Figure 6A on a decryptor

1 bitstream made up of an arbitrary number of 64-bit data  
blocks.

#### DETAILED DESCRIPTION

6 Figure 2 shows an FPGA 200, which includes  
configuration logic 205 and an array 210 of configurable  
elements. Although not shown, configurable array 210  
typically includes CLBs, interconnect lines, and IOBs  
similar to those described above in connection with Figure  
1. FPGA 200 is configured by loading one or more  
11 configuration bitstreams into internal memory cells in  
array 210 that define how the CLBs, interconnect lines, and  
IOBs of array 210 are configured. FPGA 200 also includes  
some non-volatile memory 212 adapted to include a  
decryption key.

16 In accordance with the invention, FPGA 200 is  
configured using two configuration bitstreams. The first, a  
decryptor bitstream 213, includes configuration data  
designed to instantiate a decryptor 215 in array 210. The  
second, an encrypted bitstream 216, is encrypted  
21 configuration data designed to instantiate some desired  
functional circuitry 225 in array 210. Encrypted bitstream  
216 might represent proprietary bus-interface logic, for  
example.

FPGA 200 is programmed by first supplying decryptor  
26 bitstream 213 to configuration logic 205. Configuration  
logic 205 uses decryptor bitstream 213 to instantiate a  
decryptor 215 within array 210. Encrypted bitstream 216,  
for implementing the proprietary functional circuitry, is  
then presented to an input terminal of decryptor 215.  
31 Decryptor 215 uses a pre-programmed key in non-volatile  
memory (NVM) 212 to decrypt encrypted bitstream 216 and  
present the resulting decrypted bitstream to configuration  
logic 205. Configuration logic 205 then uses the decrypted  
bitstream to instantiate proprietary functional circuitry  
36 225. Dashed arrows in Figure 2 depict the data path along  
which encrypted bitstream 216 is decrypted and instantiated

1 as functional circuitry 225.

In reference to Figure 1, a vendor might sell the general-purpose chip set 100 with some proprietary configuration data stored in ROM 110. In accordance with the invention, the proprietary data can be encrypted and  
6 FPGA 100 modified to include non-volatile memory 212 programmed with a secret decryption key. The encrypted configuration data would only work with those FPGAs programmed to include the correct key. Thieves will therefore find it very difficult to copy the configuration  
11 data.

Figure 3 shows an FPGA 300 that includes configuration logic 305, random-access memory (RAM) 310, and an array 315 of configurable logic. In accordance with the invention, FPGA 300 additionally includes hard-wired hash-function  
16 logic 320, a hash register 322, a pair of XOR gates 325 and 330, and non-volatile memory (NVM) 335. NVM 335, in turn, includes memory locations 340, 342, and 345 for storing respective encryption, hash-function, and decryption keys. NVM 335 may be, for example, conventional flash, antifuse,  
21 or mask programmed memory. Also in accordance with the invention, array 315 includes a decryptor 350 derived from a decryptor bitstream 352 and some proprietary functional circuitry 355 derived from an encrypted bitstream 357. In one embodiment, FPGA 300 is one of the Virtex™ family of  
26 FPGAs available from Xilinx, Inc.

Figure 4 is a flowchart 400 depicting the process of programming FPGA 300 of Figure 3 to include decryptor 350 and functional circuitry 355. This process is performed automatically each time FPGA 300 is powered on or reset.  
31 Beginning with step 405, decryptor bitstream 352 is presented to a designated I/O pin of FPGA 300. Configuration logic 305 uses decryptor bitstream 352 to instantiate decryptor 350 into array 315. Decryptor bitstream 352 is sent "in the clear," meaning that it is  
36 not encrypted. Transmitting decryptor bitstream 352 in the clear is not considered a breach of security because

1 cryptographers assume that everyone knows the encryption  
algorithm. The security lies in the secrecy of decryption  
key 345.

6 A clever thief might engineer an FPGA design that,  
when instantiated into array 315, simply reads decryption  
key 345 and presents the key on an output pin. To forestall  
such a security breach, FPGA 300 authenticates decryptor  
350 by performing a hash function on decryptor bitstream  
352 while configuration logic 305 instantiates decryptor  
350 (step 410). The result of the hash function, the "hash  
11 result," is stored in hash register 322 and compared to the  
proprietary hash key 342 (step 415). Only those bitstreams  
that produce the desired hash result will provide access to  
decryption key 345. In one embodiment, hash-function logic  
320 encrypts the incoming decryptor bitstream using a  
16 technique commonly known as cipher-block chaining (CBC).  
This embodiment is described below in connection with  
Figures 6A and 6B.

If in step 415 the hash result in hash register 322  
does not match hash key 342, then the incorrect key (or no  
21 key) is presented to the instantiated decryptor (step 420).  
Without access to the correct decryption key 345, any  
subsequent attempt to decrypt an incoming encrypted  
bitstream 357 will fail (step 425), resulting in a  
dysfunctional FPGA. If the hash result in hash register 322  
26 matches hash key 342, then the correct decryption key 345  
is presented to the instantiated decryptor 350 (step 430).

Encrypted bitstream 357, representing the proprietary  
functional circuitry 355, is presented to the instantiated  
decryptor 350 in the FPGA (435). With access to the correct  
31 decryption key 345, decryptor 350 will correctly decrypt  
encrypted bitstream 357 and provide the resulting decrypted  
bitstream to an input terminal of configuration logic 305.  
Finally, configuration logic 305 configures array 315 using  
the decrypted bitstream to instantiate functional circuitry  
36 350 (step 440), resulting in a functional FPGA.



1 FPGA 300 includes one example of circuitry designed to  
deny decryption-key access to unauthenticated circuits.  
Hash-function logic 320 stores the hash result from  
decryptor bitstream 352 in hash register 322. XOR gate 325  
then compares the hash result in hash register 322 with the  
6 secret hash key 342. If the hash result and hash key match,  
then XOR gate 325 outputs a logic zero to a first input  
terminal of XOR gate 330. If, on the other hand, the hash  
result in hash register 322 and hash key 342 do not match,  
then XOR gate 325 outputs a logic one to the first input  
11 terminal of XOR gate 330.

Decryption key 345 connects to the second input  
terminal of XOR gate 330. XOR gate 330 outputs decryption  
key 345 when the input terminal from XOR gate 330 is a  
logic zero, and outputs an inverted version of decryption  
16 key 345 when the input terminal from XOR gate 330 is a  
logic one. As discussed above, XOR gate 325 provides a  
logic zero to XOR gate 330 only when the hash result in  
hash register 322 matches hash key 342. Thus, XOR gate 330  
will only present the correct decryption key if the hash  
21 function of decryptor bitstream 352 matches hash key 342.

For illustrative purposes, XOR gates 325 and 330 are  
each shown to include two input terminals and one output  
terminal. However, XOR gates 325 and 330 typically include  
a number of input terminal pairs and an equal number of  
26 output terminals. In one embodiment, for example, each of  
XOR gates 325 and 330 includes 64 pairs of input terminals  
and 64 output terminals. In that embodiment, XOR gate 330  
compares a 64-bit hash result in hash register 322 with a  
64-bit hash key 342. If any bit does not match, then the  
31 corresponding output bit from XOR gate 325 will be a logic  
one. Consequently, the signal on the corresponding output  
terminal from XOR gate 330 will be logically opposite the  
appropriate decryption key bit, and the circuit  
instantiated by the bitstream that produced the incorrect  
36 hash result will not have access to the correct decryption  
key.

1 While the output terminal of hash key 342 and hash  
register 322 represent the same number of bits as  
decryption key 345, this need not be the case. In one  
embodiment, for example, the parallel output terminals of  
XOR gate 325 are ORed and the result is presented to one  
6 half the inputs to XOR gate 330. Thus configured, if any  
bit of hash key 342 does not match the output terminal of  
hash register 322, then one half of inputs to XOR gate 330  
will be logic ones. XOR gate 330 will therefore invert  
decryption key 345. Alternatively, the output terminals of  
11 the added OR gate could be fed to the inputs of a second OR  
gate substituted for XOR gate 330. In that case, a mismatch  
between hash key 342 and hash register 322 will cause all  
logic ones to be presented to decryptor 350 in lieu of the  
correct decryption key (presumably, decryption key 345 is  
16 not selected to be all ones).

FPGA 300 includes block RAM 310. Some embodiments of  
the invention take advantage of block RAM 310 by storing  
some decrypted configuration data in block RAM 310. Then,  
once decryptor 350 is no longer needed, the configuration  
21 data in block RAM 310 is used to configure the portion of  
array 315 in which decryptor 350 resided. This process  
allows for more efficient use of array 315. Alternatively,  
the portion of array 315 in which decryptor 350 resides can  
be programmed in the clear after decryptor 350 decrypts  
26 functional circuitry 355.

A DES algorithm is used, in one embodiment, to encrypt  
the bitstream used to instantiate functional circuitry 225  
(Figure 2) and functional circuitry 355 (Figure 3).  
Decryptors 215 and 350 of Figures 2 and 3 perform the  
31 inverse of the same DES function to decrypt encrypted  
bitstreams. The DES algorithm is well known to those of  
skill in cryptography. Figure 5 is a flowchart 500  
summarizing the Data Encryption Standard (DES) encryption  
algorithm. For a detailed treatment of DES, used both for  
36 encryption and decryption, see "Applied Cryptography,  
Second Edition: Protocols, Algorithms, and Source Code in

1 C," by Bruce Schneier (1996). Pages 265-285 of Schneier  
relate specifically to DES. A Xilinx application note  
entitled "DES Encryption and Decryption on the XC6216," by  
Ann Duncan (February 2, 1998), describes the design and  
implementation of DES encryption/decryption on an XC6216™  
6 FPGA available from Xilinx, Inc.

Figure 6A is a block diagram 600 representing the hash  
function performed by hash-function logic 320 of Figure 3.  
For simplicity, the bitstream in the illustrated example  
consists of three 64-bit data blocks  $B_1$ ,  $B_2$ , and  $B_3$ ; the hash  
11 function can be extended to any number and size of data  
blocks. In one embodiment, hash-function logic 320 uses a  
cipher-block chaining method outlined in the above-cited  
Schneier reference on e.g. pages 193-197.

The first data block  $B_1$  is encrypted using a  
16 conventional encryption algorithm  $E_G$ , in one embodiment the  
DES algorithm described above in connection with Figure 5.  
This encryption employs a secret encryption key "G"  
(encryption key 340 of Figure 3, for example) to encrypt  
the first data block  $B_1$ . The resulting encrypted 64-bit  
21 block  $E_G(B_1)$  is then XORed with the second data block  $B_2$ , the  
XOR function being represented by a conventional XOR symbol  
610. The resulting 64-bit value,  $\{E_G(B_1)\} \oplus B_2$ , is then XORed  
with the next data block  $B_3$  and the result is subjected to  
the encryption algorithm  $E_G$  to produce the hash value. This  
26 process, conventionally known as cipher block chaining,  
produces a 64-bit hash value 615 that depends upon all of  
data blocks  $B_{1-3}$ .

Figure 6B is a flowchart 650 illustrating a method of  
performing the hash function of Figure 6A on a decryptor  
31 bitstream made up of an arbitrary number of 64-bit data  
blocks. This method is implemented by hash-function logic  
320 of Figure 3 in one embodiment of the invention.

In step 655, hash-function logic 320 encrypts the  
first 64-bit data block of an incoming decryptor bitstream  
36 and stores the resulting encrypted data in hash register  
322 (i.e.,  $R_{322}$ ). Then, for each additional block  $B_N$ , hash-

1 function logic 320:

1. performs a 64-bit exclusive OR (XOR) of the contents of register 322 and the additional block  $B_N$ ;
2. encrypts the contents of hash register 322 using encryption key G; and
3. stores the result,  $E_G(R_{322} \oplus B_N)$ , back in hash register 322.

The foregoing procedures are represented in Figure 6B as the "For" loop that includes steps 660, 665, and 670.

11 When no more data blocks are available (e.g., when hash-function logic 320 reaches the end of the decryptor bitstream 352), hash register 322 contains the hash value of decryptor bitstream 352. As discussed above in connection with Figure 3, XOR gate 325 compares hash value  
16 in hash register 322 with hash key 342 to ensure that decryptor bitstream 352 represents an authorized decryptor. If not, then XOR gate 330 presents the wrong decryption key to instantiated decryptor 350.

Some PLDs are designed to respond to a "readback" command by outputting a bitstream (the readback data) that includes the configuration data of the PLD. The readback command is disabled on devices implementing the present invention to prevent a thief from simply reading back the decrypted configuration data. Alternatively, an encryptor  
26 could be instantiated on a PLD to re-encrypt configuration data readout of the PLD. For more information relating to readback operations on Xilinx XC4000™ series FPGAs, see Xilinx, Inc., "The Programmable Logic Data Book" (1998), pp. 4-56 to 4-59, and Wolfgang Höflich, "Using the XC4000™  
31 Readback Capability," XAPP 015.000, pp. 8-37 to 8-44 (1993). Both of these documents are available from Xilinx, Inc., of San Jose, California.

Various nodes within FPGA 300 must be protected from observation to avoid compromising security. These nodes  
36 include the output terminals of proprietary keys 340, 342, and 345 of NVM 335 and the output terminal of decryptor

1 350. Care should therefore be taken to ensure that such  
nodes are not and cannot be configured to be accessed via  
any input/output pins of FPGA 300.

Some configuration information is easily observed once  
the FPGA is operational. For example, one can measure the  
6 voltage on an input/output block of an FPGA to determine  
whether that input/output block is configured to include a  
pull-up resistor. If this observable data is a result of  
some decryption, skilled cryptologists can make use of this  
data to learn something about the decryption process, and  
11 possibly to breach security. It may be desired, therefore,  
to identify those bits of configuration data that can be  
easily observed once the FPGA is configured and to transmit  
those data in the clear. Of course, the encryptor and  
decryptor must both understand which data is to be  
16 transmitted in the clear and which is to be encrypted.

Hash-function logic 320 and decryptors 215 and 350 are  
not limited to the DES algorithm; other types of algorithms  
-- many of which are well known -- can also be used. For  
example, a public-key algorithm such as RSA (named for its  
21 creators - Rivest, Shamir, and Adleman) can be used for both  
encryption and decryption. FPGA vendors could then program  
a private key into non-volatile memory on the FPGA and core  
developers could use a corresponding public key to encrypt  
their designs. Moreover, several decryption keys can be  
26 stored in each FPGA so that a different key can be used in  
the event that one of the keys is stolen.

Configuring an FPGA to include a decryptor, as opposed  
to fabricating the FPGA with a hard-wired decryptor, saves  
valuable die area and allows users to select appropriate  
31 encryption/decryption schemes. For example, some desirable  
algorithms are not approved for export. A user may  
therefore select an approved decryptor for export and  
select another algorithm for local sale. Alternatively, a  
distributor of FPGAs can simply sell standard FPGAs and  
36 allow purchasers to select the appropriate legal decryption  
scheme that provides a desired level of security.

1           While the present invention has been described in  
connection with specific embodiments, variations of these  
embodiments will be obvious to those of ordinary skill in  
the art. For example,

1.       some FPGAs might be programmed with additional  
6 keys to support multiple decryptors or hash functions;
2.       the decryptor and encrypted bitstreams can be  
combined into a single bitstream;
- 3.

11           decryption and hash keys could be implemented using  
digital logic integrated with other PLD circuits to make  
the key values more difficult to discover by reverse  
engineering (e.g., a decryption key could be nodes of a  
logic circuit integrated into the decryptor).

16           Moreover, some components are shown directly connected to  
one another while others are shown connected via  
intermediate components. In each instance the method of  
interconnection establishes some desired electrical  
communication between two or more circuit nodes, or  
terminals. Such communication may often be accomplished  
21 using a number of circuit configurations, as will be  
understood by those of skill in the art. Therefore, the  
spirit and scope of the appended claims should not be  
limited to the foregoing description.

1     CLAIMS

1.   A programmable logic device comprising:
  - a.   an input pin adapted to receive encrypted configuration data;
  - 6       b.   a non-volatile memory element adapted to store a decryption key;
  - c.   a decryptor having a first input terminal adapted to receive the encrypted configuration data, a second input terminal adapted to access  
11       the decryption key, and an output terminal, wherein the decryptor is adapted to decrypt the encrypted configuration data and to provide resulting decrypted configuration data on the output terminal;
  - 16       d.   an array of configurable logic; and
  - e.   configuration logic having an input terminal connected to the decryptor output terminal and an output terminal connected to the array, the configuration logic being adapted to receive the  
21       configuration data and to configure the array as directed by the configuration data.
2.   The programmable logic device of Claim 1, wherein at  
26       least a portion of the decryptor is instantiated in the array of configurable logic.
3.   The programmable logic device of Claim 2, further comprising hash-function logic adapted to authenticate the portion of the decryptor.  
31
4.   The programmable logic device of Claim 3, further comprising a second non-volatile memory element connected to the hash-function logic, the second non-volatile memory element adapted to store a hash key.  
36

1        5.    The programmable logic device of Claim 3, further  
         comprising a second non-volatile memory element  
         connected to the hash-function logic, the second non-  
         volatile memory element adapted to store an encryption  
         key.

6

6.    A programmable logic device comprising:  
      a.    non-volatile memory adapted to include a secret  
             key;  
      b.    an array of programmable logic configured to  
11       include a decryptor, the decryptor including:  
             i.    an input terminal adapted to receive  
                 encrypted configuration data; and  
             ii.   an output terminal adapted to provide a  
                 decrypted version of the encrypted  
16       configuration data; and  
      c.    configuration logic adapted to receive the  
             decrypted version of the encrypted configuration  
             data.

21       7.    The programmable logic device of Claim 6, further  
         comprising a plurality of pins adapted to provide  
         electrical access to and from the programmable logic  
         device from circuits external to the programmable  
         logic device, wherein the output terminal of the  
26       decryptor is not connected to any one of the pins.

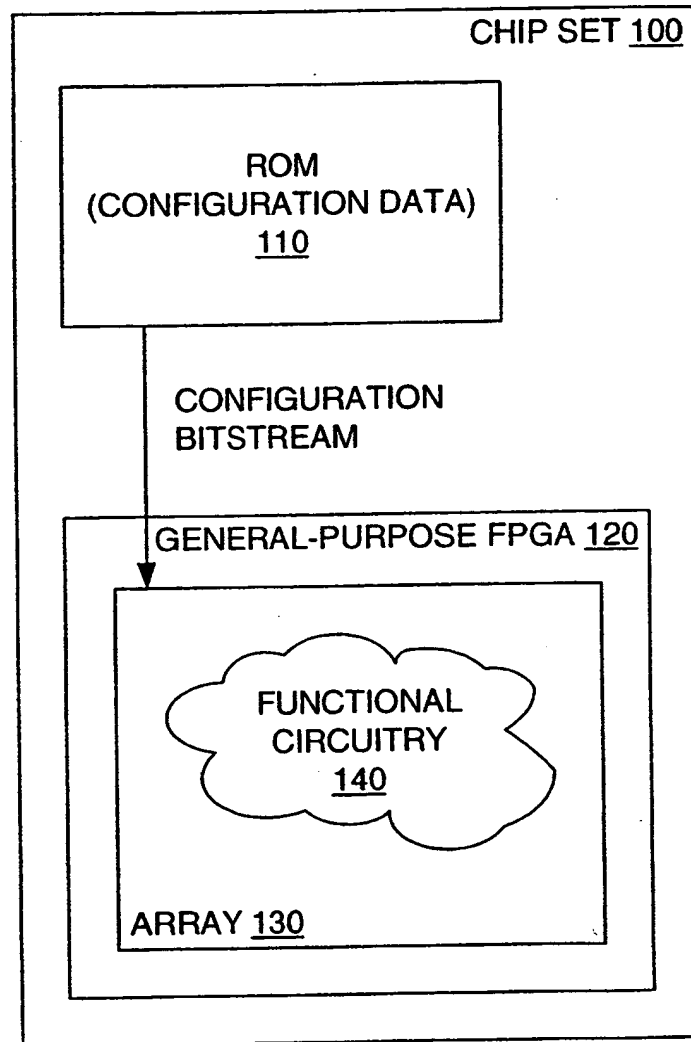
8.    A method of configuring a programmable logic device to  
         perform a desired logic function, the method  
         comprising:  
31       a.    configuring the programmable logic device to  
             include a decryptor;  
             b.    sending encrypted configuration data to the  
                 decryptor;  
             c.    decrypting the encrypted configuration data to  
36       produce decrypted configuration data  
             representing the desired logic function; and



- 1           d.     configuring the programmable logic device to  
              perform the desired logic function using the  
              decrypted configuration data.
- 6           9.     The method of Claim 8, further comprising removing the  
              decryptor after decrypting the encrypted configuration  
              data.
- 11          10.    The method of Claim 8, wherein configuring the  
              programmable logic device to include a decryptor  
              comprises providing a bitstream representing the  
              decryptor to the programmable logic device.
- 16          11.    The method of Claim 10, wherein configuring the  
              programmable logic device to include a decryptor  
              further comprises performing a hash function on the  
              bitstream representing the decryptor to authenticate  
              the decryptor.
- 21          12.    The method of Claim 11, wherein performing the hash  
              function produces a hash result, the method further  
              comprising comparing the hash result with a hash key  
              to authenticate the decryptor.
- 26          13.    The method of Claim 12, further comprising providing  
              the decryptor access to a decryption key only if the  
              hash result matches the hash key.
- 31          14.    A system comprising:  
              a.     a programmable logic device having an input  
                     terminal; and  
              b.     a memory having an output terminal connected to  
                     the input terminal of the programmable logic  
                     device, the memory programmed to include:  
                     i.     decryptor data adapted to instantiate a  
36                       decryptor in the programmable logic  
                     device; and

- 1                   ii.    encrypted configuration data adapted to  
                    instantiate a desired logic function in  
                    the programmable logic device.
- 6           15.    A system for protecting configuration data adapted to  
                  instantiate a desired logic function in a programmable  
                  logic device, the system comprising:  
                  a.    means for encrypting the configuration data;  
                  b.    means for configuring the programmable logic  
                  device to include a decryptor;  
11           c.    means for sending the encrypted configuration  
                  data to the decryptor to produce decrypted  
                  configuration data; and  
                  d.    means for instantiating the desired logic  
                  function using the decrypted configuration data.
- 16           16.    The system of Claim 15, further comprising means for  
                  removing the decryptor after configuring the  
                  programmable logic device to perform the desired  
                  function.
- 21           17.    The system of Claim 15, further comprising means for  
                  authenticating the decryptor.

1/5



**FIG. 1**  
(PRIOR ART)

2/5

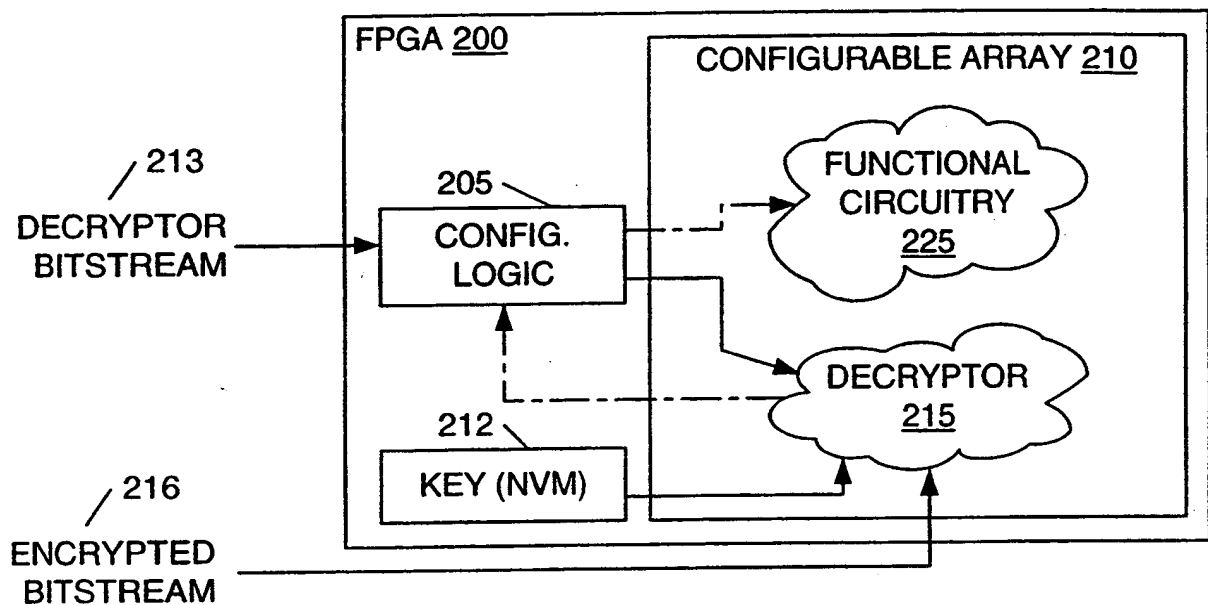


FIG. 2

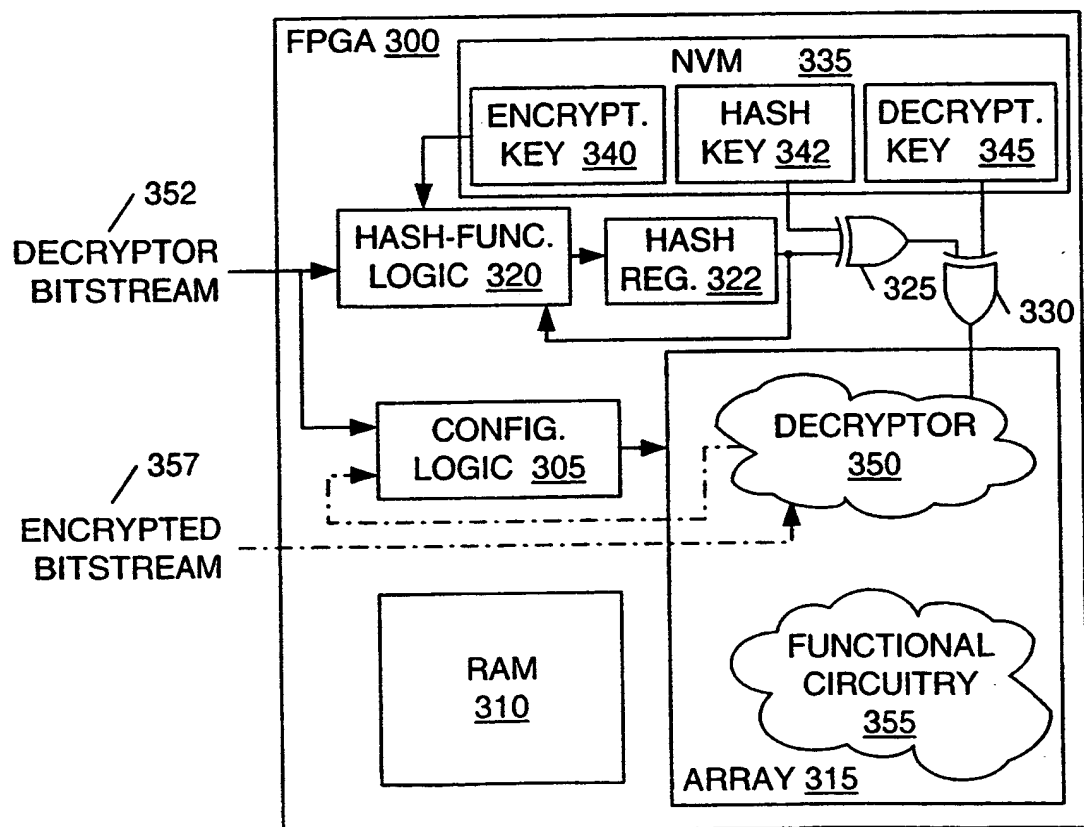


FIG. 3

3/5

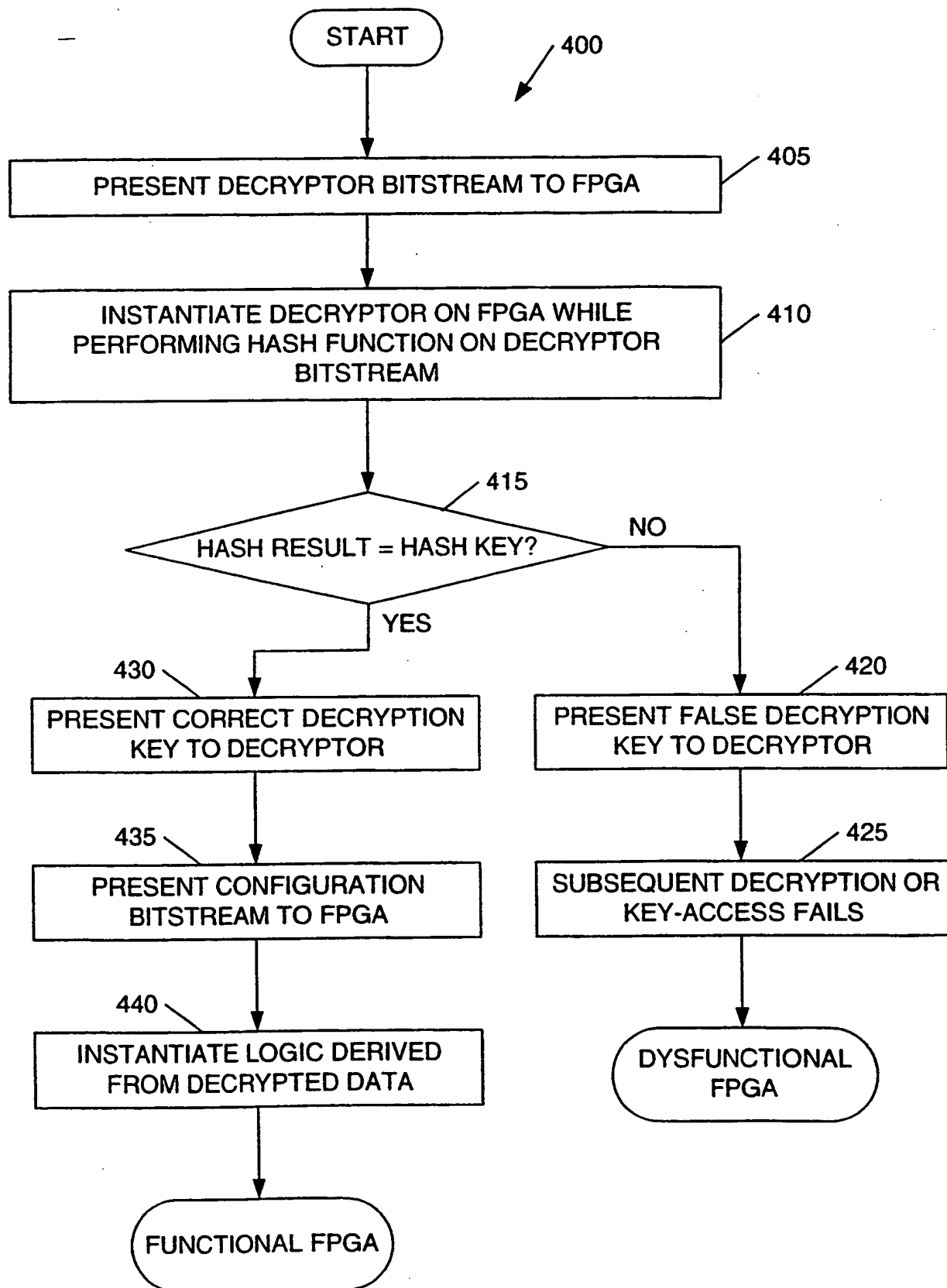


FIG. 4

4/5

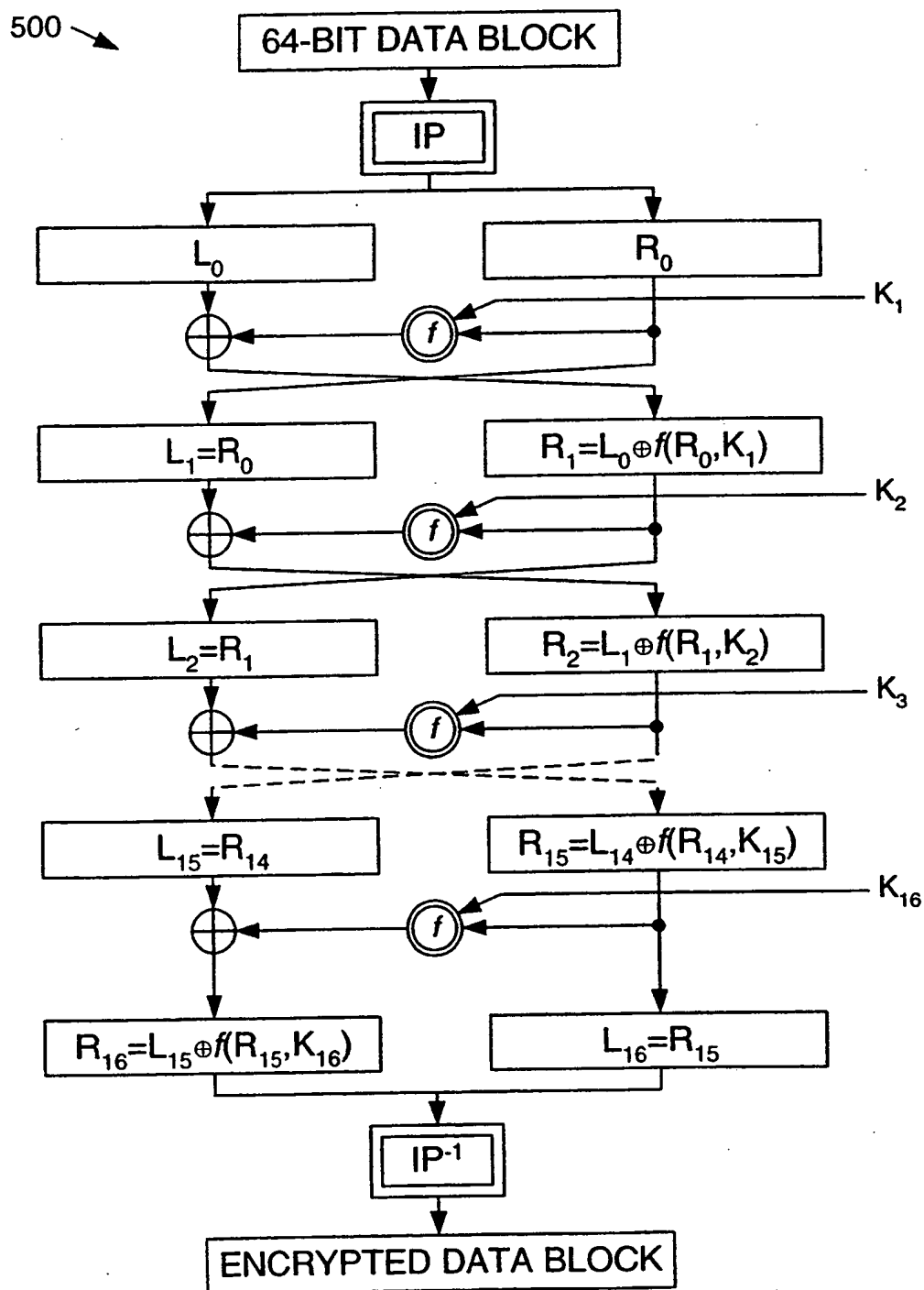


FIG. 5  
(PRIOR ART)

5/5

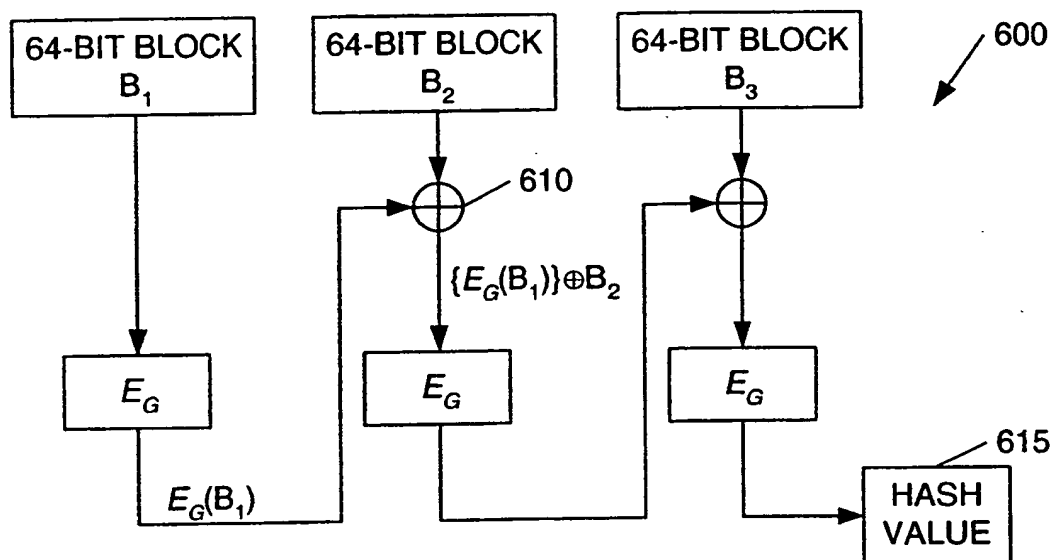


FIG. 6A  
(PRIOR ART)

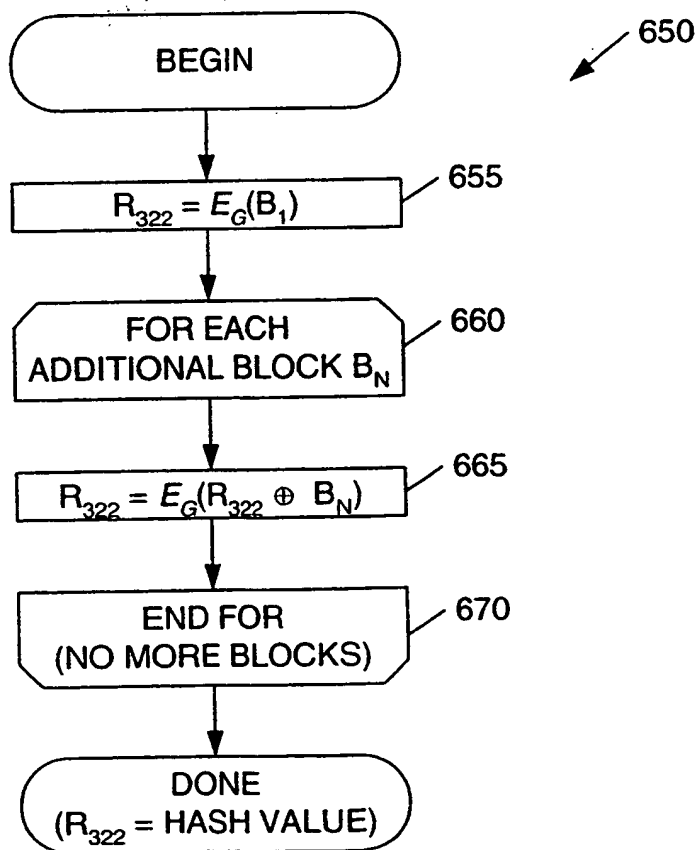


FIG. 6B  
(PRIOR ART)

**THIS PAGE BLANK (USPTO)**

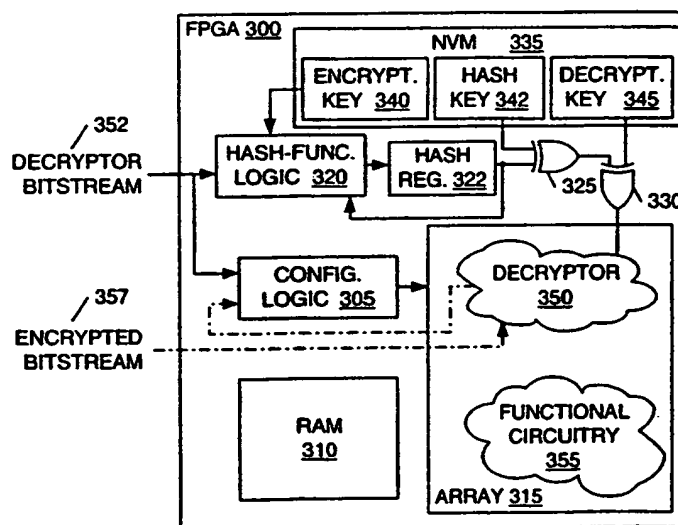




## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<b>(51) International Patent Classification <sup>7</sup> :</b> <b>H03K 19/177, H04L 9/00</b>	<b>A3</b>	<b>(11) International Publication Number:</b> <b>WO 00/49717</b> <b>(43) International Publication Date:</b> 24 August 2000 (24.08.00)
<b>(21) International Application Number:</b> PCT/US00/01398 <b>(22) International Filing Date:</b> 19 January 2000 (19.01.00) <b>(30) Priority Data:</b> 09/253,401 19 February 1999 (19.02.99) US <b>(71) Applicant:</b> XILINX, INC. [US/US]; 2100 Logic Drive, San Jose, CA 95124 (US). <b>(72) Inventor:</b> TRIMBERGER, Stephen, M.; 1261 Chateau Drive, San Jose, CA 95120 (US). <b>(74) Agents:</b> YOUNG, Edel, M. et al.; Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 (US).		<b>(81) Designated States:</b> JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).  <b>Published</b> <i>With international search report.</i>  <b>(88) Date of publication of the international search report:</b> 2 November 2000 (02.11.00)

**(54) Title:** METHOD AND APPARATUS FOR PROTECTING PROPRIETARY CONFIGURATION DATA FOR PROGRAMMABLE LOGIC DEVICES

**(57) Abstract**

Described are a method of programming a programmable logic device using encrypted configuration data and a programmable logic device (PLD) adapted to use such encrypted data. A PLD (300) is adapted to include a decryptor (350) having access to a non-volatile memory element (335) programmed with a secret decryption key (345). Some or all of the decryptor (350) can be instantiated in configurable logic on the FPGA (300). Encrypted configuration data (357) representing some desired circuit functionality is presented to the decryptor (350). The decryptor (350) then decrypts the configuration data (357), using the secret decryption key (345), and configures the FPGA (300) with the decrypted configuration data. Some embodiments include authentication circuitry that performs a hash function on the configuration data (352) used to instantiate the decryptor (350) on the PLD (300). The result (322) of the hash function is compared to a proprietary hash key (342) programmed into the PLD (300). Only those configuration data (352) that produce the desired hash result (322) will instantiate decryptors (350) that have access to the decryption key (345).

**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

# INTERNATIONAL SEARCH REPORT

International Application No:  
PCT/US 00/01398

## A. CLASSIFICATION OF SUBJECT MATTER

H03K19/177,H04L9/00

According to International Patent Classification (IPC) or to both national classification and IPC<sup>2</sup>

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H03K19/00,H04L9/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5388157 A (AUSTIN) 07 February 1995, abstract, claims, fig.	1
A	---	2-17
A	US 5768372 A (SUNG ET AL.) 16 June 1998, abstract, figs.	1-17
A	US 5349249 A (CHIANG ET AL.) 20 September 1994, abstract, figs.	1, 6, 8, 14, 15

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

### \* Special categories of cited documents:

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier document but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

- "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- "&" document member of the same patent family

Date of the actual completion of the international search

19 June 2000

Date of mailing of the international search report

04 08 2000

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

FELLNER

**ANHANG**

Zum internationalen Recherchenbericht über die internationale Patentanmeldung Nr.

In diesem Anhang sind die Mitglieder der Patentfamilien der im obengenannten internationalen Recherchenbericht angeführten Patentdokumente angegeben. Diese Angaben dienen nur zur Unterrichtung und erfolgen ohne Gewähr.

**ANNEX**

To the International Search  
Report to the international Patent  
Application No.

PCT/US 00/01398 SAE 273330

This annex lists the patent family members relating to the patent documents cited in the above-mentioned search report. The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

**ANNEXE**

Au rapport de recherche international relatif à la demande de brevet international n°

La présente annexe indique les membres de la famille de brevets relatifs aux documents de brevets cités dans le rapport de recherche international visée ci-dessus. Les renseignements fournis sont donnés à titre indicatif et n'engagent pas la responsabilité de l'Office.

Im Recherchenbericht angeführte Patentdokumente Patent document cited in search report Document de brevet cité dans le rapport de recherche			Datum der Veröffentlichung Publication date Date de publication		Mitglied(er) der Patentfamilie Patent family member(s) Membre(s) de la famille de brevets		Datum der Veröffentlichung Publication date Date de publication	
US A	5388157	07-02-1995	AT	E	182016		15-07-1999	
			CA	AA	2080299		12-04-1993	
			DE	C0	69229531		12-08-1999	
			DE	T2	69229531		23-03-2000	
			EP	A2	536943		14-04-1993	
			EP	A3	536943		16-06-1993	
			EP	B1	536943		07-07-1999	
			GB	A0	9121591		27-11-1991	
			GB	A0	9220584		11-11-1992	
			GB	A1	2260431		14-04-1993	
			GB	B2	2260431		14-06-1995	
			JP	A2	5257678		08-10-1993	
US A	5768372	16-06-1998	US	A	5915017		22-06-1999	
US A	5349249	20-09-1994	EP	A1	654190		24-05-1995	
			JP	T2	8501402		13-02-1996	
			WO	A1	9423500		13-10-1994	